REGMPE

Revista de Empreendedorismo e Gestão de Micro e Pequenas Empresas

USE OF OPERATIONAL RESEARCH IN THE MANAGEMENT OF A MICRO ENTERPRISE

Yudi Matsuguma Yoshida[1]
Stella Jacyszyn Bachega[2]
Dalton Matsuo Tavares[3]

abstract

The search for the growth of a company with the lowest costs is one of the biggest challenges for managers today. One of several options is the use of optimization techniques present in Operations Research. There are several optimizers based on programming languages aimed at solving optimization problems. However, each language has its particularities and, consequently, its limitations. The Julia language was created with the aim of combining the advantages of several languages with a relatively simple syntax, which makes it more user friendly. The purpose of this article is to apply Operational Research techniques to assist the management of a microenterprise. For this, the quantitative approach and the experimental research procedure were used. With the proper translations of the problems into the Julia language,*mix*production of snacks for a food company and the shortest route for their delivery. By addressing linear programming techniques and the traveling salesman problem, this article can be explored as an instructive tool in future scientific research and / or to aid in an organization's decision-making process.

Key words: Operational Research. Julia language. Linear Programming. Traveling Salesman Problem.

1. Introduction

Operational Research (PO) is a research of operations, which is used in problems that require management, coordination and decision making in relation to the operations of an organization (HILLIER; LIEBERMAN, 2013). Therefore, PO can be approached in several segments, such as production, logistics, financial management and production planning and control, which denotes its interdisciplinarity and multidisciplinarity (ARENALES et al., 2007).

Several of these approaches, which required a large volume of accounts and which were often not feasible to be solved manually, could be solved with the advancement of computer technology and the creation of several specialized optimization software (MOREIRA, 2010). Such programs have advantages and disadvantages in relation to productivity and performance.

[1]Federal University of Goiás - Regional Catalão - Special Academic Engineering Unit (FENG). E-mail: yudimatsuguma@gmail.com. ORCID: https://orcid.org/0000-0002-5701-7837
[2]Federal University of Goiás - Regional Catalão - Special Academic Engineering Unit (FENG). E-mail: stella@ufg.br. ORCID: https://orcid.org/0000-0002-7533-5361
[3]Federal University of Goiás - Catalan Regional - Special Academic Unit of Biotechnology (IBIOTEC). E-mail: dalton_tavares@ufg.br. ORCID: https://orcid.org/0000-0001-8531-5578

As an example, the C and C ++ languages, which can be used in PO studies, allow the implementation of algorithms for a quick execution, but they have a complex syntax that requires specific computer skills. The Phyton language, on the other hand, contains several high-level functions and tools, which offers the execution of algorithms with few lines of code. On the other hand, its performance is inferior to C / C ++ (JULIALANG.ORG, 2018).

In order to replace several languages, in order to combine its advantages with a relatively simple syntax, the Julia language appeared in 2009 at the Massachusetts Institute of Technology (MIT). This was based on several languages in an attempt to keep it generic, similar to C, Fortran and Phyton, with the aim of being used by programmers, mathematicians, statisticians and researchers (BEZANSON et al., 2012).

Even though the language is relatively 'young', there are already several scientific works published in different areas and disciplines, such as genetics, linear algebra, statistics, econometrics, computational neuroscience, risk analysis and also, PO. Based on this context, the present research is justified by the importance of the theme, as can be seen in recent publications that used the Julia language, as in Anderson et al. (2017), Baldassi (2017), Bezanson et al. (2017), Castelluccia (2017), Mogensen and Riseth (2018), Rackauckas et al. (2018). Still, authors like Almeida et al. (2017), Santana et al. (2018), Sousa et al. (2019), Souza et al. (2019) and Teixeira et al. (2017) encouraged the application of PO techniques in micro and small companies to improve business management. Like this,

Thus, the general objective of the article is to apply Operational Research techniques to assist the management of a micro company operating in the food industry. For that, the Julia language was used. Through the analysis of the business reality, a Linear Programming (PL) problem was selected in order to maximize the company's revenue by determining the best mix of snacks, as long as their restrictions are respected. In addition, the delivery of the snacks is made by the company itself, and there is also the opportunity to apply the traveling salesman problem.

The structure of the article is as follows: section two presents the theoretical framework and section three discusses the approach of the methodology used to carry out the work. Section four deals with the results and discussions. Finally, in the last section are the final considerations.

2. Theoretical framework

2.1 Julia

Created in 2009 at MIT, the Julia programming language is considered to be high-level, dynamic and has high performance for computer graphics and numerics. It has a sophisticated compiler, distributed parallel execution, numerical precision and an extensive library of mathematical functions. In addition, it has a generic syntax based on several languages (C, Fortran, Matlab, Phyton, among others), so that the software can be used not only by programmers, but also by mathematicians, statisticians and researchers (BEZANSON et al., 2012).

According to the official website, programming has the following main characteristics: (i) multimethods, where it has the ability to define the behavior of the function through various combinations of types of arguments; (ii) parallelism and distributed computing capacity, making it possible to divide the problem into parts and distribute it among several computers; (iii) has a practical and simple to use package manager; (iv) good performance, bringing it closer to the C language in several aspects; and (v) powerful ability to manage other processes (JULIALANG.ORG, 2018).

2.2 Operational research: definition and some techniques

According to Hillier and Lieberman (2013), PO is about researching operations, being used in problems that require management, coordination and decision making regarding operations in an organization. Therefore, several different areas (such as logistics, financial management, production planning and control, health, among others) may have their problems solved using PO techniques.

One of the most general techniques used in PO is systems simulation, which consists of trying to accurately represent a real system in a computer system, allowing hypothesis testing of new scenarios and, consequently, assisting in the organization's decision-making process (PEREIRA et al., 2015). The introduction of a new fuel pump at a gas station is an example of a simulation study.

Linear Programming (PL) is responsible for solving the problems of allocating limited resources to activities that compete with each other for such resources. However, PL has other applications, as long as the problem formulation fits in with its generic mathematical format (HILLIER; LIEBERMAN, 2013).

The problems that have a relationship between their components, so that they can be formulated as a network, are solved through network optimization. Its use is very wide since networks exist in several areas, such as transport, electrical, communications, production and project management networks (ARENALES et al., 2007).

Within network optimization, there are several algorithms that help in solving problems and making decisions, such as: minimum spanning tree algorithm, whose objective is to connect all nodes in the network, directly or indirectly, so that the total length of arches are as small as possible; maximum flow algorithm, which aims to determine the route of passage from the origin to the destination node (sink) with the highest total amount of flow possible; critical path algorithm, widely used in project management in determining critical project activities; shortest path problem, which deals with determining the path that has the shortest length between the source node and the destination node; and the traveling salesman problem (TAHA, 2008).

2.3 Linear programming

Linear Programming (PL) emerged in the late 1940s and, with the creation of the computer in the following decade, its development was accelerated and widespread (PRADO, 2007). For Moreira (2010), PL is one of the most popular mathematical models, configured to solve problems that have measurable and correlated variables, expressed by mathematical equations and / or inequalities. The popularization of the model is due to the fact that there are numerous problems in the scientific and social areas that can be formulated in this way.

According to Ehrlich (1985), PL is a planning tool that helps in deciding which activities (decision variables) to undertake, since they compete with each other for the use of scarce resources (restrictions), since such resources are insufficient to that all activities are carried out at the maximum level desired.

Today, it has become a standard tool that has saved millions of dollars for many companies. It is worth mentioning that PL is not restricted to the industrial sector only, but also to various areas of society (HILLIER; LIEBERMAN, 2013).

According to Passos (2008), PL is an optimization technique applied to a system of equations, or inequalities, linear mathematics of a problem. Its objective is to maximize or minimize a linear function by determining the values of the decision variables, taking into account that the restrictions must be satisfied.

According to Passos (2008), the model pattern of an LP problem is given according to Eq. 1 to 7.

$$\text{Max (or Min) } Z = c1x1 + c2x2 + c3x3 + ... + cnxn \tag{1}$$

$$\text{Subject to: } a11x1 + a12x2 + a13x3 + ... + a1nxn = b1 \tag{2}$$

$$a21x1 + a22x2 + a23x3 + ... + a2nxn = b2 \tag{3}$$

$$a31x1 + a32x2 + a33x3 + ... + a3nxn = b3 \tag{4}$$

$$\ldots \ldots \ldots \qquad \ldots \ldots \tag{5}$$

$$amnx1 + am2x2 + am3x3 + \ldots + amnxn = bm \tag{6}$$

$$x1 \geq 0, x2 \geq 0, x3 \geq 0, \ldots, xn \geq 0 \tag{7}$$

Where X: {x1, x2, x3, ..., xn}: are the decision variables; A: {a1, a2, a3, ..., a_m}: are the coefficients of the variables; and B: {b1, b2, b3, ..., bm}: are the independent terms that represent the available resources.Decision variables are essential for solving the problem. The objective function, given by equation (1), wants to optimize the resources involved by maximizing or minimizing them. The restrictions, given according to Eq. 2 to 7, are the limitations of the problem, such as, for example, the productive capacity of the system, quantity of labor, raw material or stock (LINS; CALÔBA, 2010, p. 7).

For linearity to be satisfied, three basic properties must be met by the problem structure: proportionality, additivity and certainty. The first property proposes that the contribution of each decision variable is directly proportional to the value of the variable, both in the objective function and in the restrictions. The second property states that the total contribution of the variables of the objective function and of the restrictions is the sum of the individual contributions of each variable. Finally, the third property concerns all the coefficients of the objective function and the restrictions of the PL model, which are known constants (TAHA, 2008).

2.4 Traveling salesman problem

The traveling salesman (PCV) problem is a combinatorial optimization problem, therefore, it is widely used in experiments of various optimization methods, since it is easy to describe, understand and has wide applicability (KARP, 1975). According to Hillier and Lieberman (2013), the PCV can be described as a problem in defining the shortest route (in terms of distance or costs) of a seller who travels several cities (nodes) through a network and, in the end, returns to the city of origin. It is worth mentioning that each city can only be visited once.

Such a problem can be solved by linear programming, more specifically, by the Simplex method, in which all possible routes for the problem will be analyzed. However, from a certain number of nodes in the problem, it is impossible to solve it by this method. Considering that "n" is the number of cities and that the PCV is symmetrical, that is, when the path from city A to city B is equal to the path from city B to city A, the number of possible routes is given by (n-1)! / 2, where there are (n-1) possibilities for the first city, (n-2) for the second and so on. The division by 2 indicates that each route has a route with the same module, but in the opposite direction. If the PCV is asymmetric, the number of possible routes is given by (n-1) !. Therefore, a problem with only 10 cities has 181.

Since G = (N, E) is a graph, where N = {1, 2, ..., n} is the set of nodes, E = {1, 2, ..., m} is the set of edges of G , the objective is to determine the smallest Hamiltonian cycle of the graph G (ARENALES et al., 2007).

It is possible to notice that depending on the value of “n”, solving the problem by the exact methods becomes impracticable. In the face of such difficulties, several heuristic approaches have been applied and provided good quality solutions, however they do not guarantee the optimality of a result, since it is determined by intuitive methods (SILVEIRA, 2000).

A well-known heuristic method is the "nearest neighbor", which selects the shortest route to the neighboring nodes the clerk is on. Therefore, the node closest to the source node is defined as the second node and, from there, it selects the closest node among the non-selected nodes, which will be the third node. Repeat the process until all nodes are connected (HILLIER; LIEBERMAN, 2013).

3. Methodology

The scientific explanation used in this research was hypothetical-deductive, according to the considerations of Carvalho (2000). Thus, the proposition that it is possible to apply PO techniques in a microenterprise to improve business management was corroborated at the end of the research.

Still, the quantitative approach according to Creswell (2010) was used, due to the way the data were manipulated in the research. As a result, mathematical models were generated with structured equations and inequalities, incorporating causal guidelines and identifying the correlation of multiple variables.

The experimental research procedure was adopted. This is more suitable for quantitative approaches and can be related to mathematical modeling and computational simulations (BRYMAN, 1989). Gil (2009) adds that this procedure covers both the determination of the object of study and the selection of variables that influence it.

The steps for conducting the PO study were supported by Hillier and Lieberman (2013). The steps performed in this research were:

Step 1) Definition of the problem of interest and data collection: in this step, classical problems of operational research were identified in order to select the problem to be modeled and the data for mathematical modeling were gathered. Linear programming and traveling salesman problems were selected. The data to support the examples present in this work were obtained from a micro-entrepreneur in the food industry.

Step 2) Formulation of the mathematical model that represents a given problem: here the mathematical model was formalized in functions, equations and inequalities, according to the appropriate PO technique for representation and later resolution.

Step 3) Development of the computational procedure to find a solution to the modeled problem: the mathematical model was translated into the Julia language and the optimizers employed determined a possible optimal solution.

Step 4) Model testing and improvement: there was a concern to validate the models, making them faithfully represent reality. Thus, the structure of the model was checked again and it was possible to find the right answer to their respective problems.

Step 5) Documentation: the documentation was made to favor the understanding of the research carried out and to guarantee its future continuity.

4. Results obtained

In this section, there is the presentation of the results obtained first in the linear programming problem and, later, in the traveling salesman problem.

4.1 Linear Programming

Linear programming (PL) is defined as an optimization problem where the objective function and restrictions are represented by linear equations, or inequalities. In addition, PL is responsible for resolving limited resource allocation problems for activities that compete with each other for such resources. As the problem defined for carrying out the study has the objective of determining the best mix of snacks that maximizes the company's revenue, based on the quantities of raw materials available in stock, production capacity and average time spent per product and that the function objective and restrictions can be represented in a linear way, it is evident that it can be treated as an LP problem. Furthermore, the variables will be treated as integers; thus, it is possible to apply the Integer Linear Programming (PLI).

The savory company, located in a city in southeastern Goiás, wants to maximize its revenue by selling six types of savory. Information regarding each salty can be found in Table 1.

Table 1 - Data on revenue from the sale of snacks

Salty	Variable (unit)	Revenue (R $) per unit
Coxinha	x1	2.50
Sausage roll	x2	2.50
Meat pastry	x3	2.50
Ham and cheese pastry	x4	2.50
Meat kebab	x5	2.50
Cheese kebab	x6	2.50

Each salty is made up of different amounts of raw materials (Table 2). As can be seen, a coxinha (quantity of coxinhas is represented by variable x1) consumes approximately 25g of potato and 65g of chicken. It should be noted that the raw material items represented in these restrictions were only those made available by the entrepreneur, as they represent a higher cost per item.

Table 2 - Quantity of raw material used for each salty and available in stock

Resource	x1	x2	x3	x4	x5	x6	Stock (g)
Potato (g)	25	0	0	0	0	0	2,500
Meat (g)	0	0	40	0	75	65	28,000
Chicken (g)	65	0	0	0	0	0	6,000
Corn (g)	0	0	20	0	0	0	4,000
Mozzarella (g)	0	35	0	50	0	25	10,000
Ham (g)	0	0	0	50	0	0	3,000
Sausage (g)	0	100	0	0	0	0	4,000
Wheat (g)	0	0	0	0	65	50	16,000

In addition, each salty has its minimum and maximum daily demand based on production capacity, as can be seen in Table 3. Each salty must have a minimum daily production of four items to meet the minimum demand. The maximum production, for example of coxinha, should be 90 units.

Table 3 - Minimum and maximum demand for each type of salty

salty	Minimum demand	Maximum demand
Coxinha	$x1 \geq 4$	$x1 \leq 90$
Sausage roll	$x2 \geq 4$	$x2 \leq 40$
Meat pastry	$x3 \geq 4$	$x3 \leq 180$
Ham and cheese pastry	$x4 \geq 4$	$x4 \leq 60$
Meat kebab	$x5 \geq 4$	$x5 \leq 160$
Cheese kebab	$x6 \geq 4$	$x6 \leq 120$

Finally, savory snacks take about 25 seconds to assemble, regardless of type. The total time reserved for the assembly of the snacks is 4 hours and 30 minutes, or 16,200 seconds. Thus, Eq. 8 refers to the last restriction of the problem.

$$25x1 + 25x2 + 25x3 + 25x4 + 25x5 + 25x6 \leq 16{,}200 \quad (8)$$

Therefore, it is possible to represent the company's problem by linear mathematical equations / inequalities. Eq. 9 refers to the objective function, while Eq. 10 to Eq. 32 are the restrictions.

$$\text{Max } Z = 2.5x1 + 2.5x2 + 2.5x3 + 2.5x4 + 2.5x5 + 2.5x6 \quad (9)$$

Subject to:

$$25x1 \leq 2500 \quad (10)$$

$$40x3 + 75x5 + 65x6 \leq 28{,}000 \quad (11)$$

$$65x1 \leq 6{,}000 \quad (12)$$

$$20x3 \leq 4{,}000 \quad (13)$$

$$35x2 + 50x4 + 25x6 \leq 10{,}000 \quad (14)$$

$$50x4 \leq 3{,}000 \quad (15)$$

$$100x2 \leq 4{,}000 \quad (16)$$

$$65x5 + 50x6 \leq 16{,}000 \quad (17)$$

$$x1 \leq 90 \quad (18)$$

$$x2 \leq 40 \quad (19)$$

$$x3 \leq 180 \quad (20)$$

$$x4 \leq 60 \quad (21)$$

$$x5 \leq 160 \quad (22)$$

$$x6 \leq 120 \quad (23)$$

$$x1 \geq 4 \quad (24)$$

$$x2 \geq 4 \quad (25)$$

$$x3 \geq 4 \quad (26)$$

$$x4 \geq 4 \quad (27)$$

$$x5 \geq 4 \quad (28)$$

$$x6 \geq 4 \quad (29)$$

$$25x1 + 25x2 + 25x3 + 25x4 + 25x5 + 25x6 \leq 16{,}200 \quad (30)$$

$$x1, x2, x3, x4, x5, x6 \geq 0 \quad (31)$$

$$\text{integers } x1, x2, x3, x4, x5, x6 \quad (32)$$

Having defined the mathematical model, the next step was to translate it into the Julia language, where the optimizer will find a possible solution. First, it was necessary to install the Julia for Mathematical Programming (JuMP) package to make Julia able to read optimization problems. It is a mathematical language used to solve linear, non-linear, mixed integer, conical, second order and semi-defined programming problems. However, the package is not able to solve optimization problems alone, so optimizers must also be installed together to interact with JuMP. In this case, in particular, the "Cbc" optimizer was installed, which is free and supports both PL and mixed integer programming problems.

Installing packages requires only two command lines, one line for each package. It is recommended that, at the end of the installation, they are updated to the latest version using another command line. Then, it is necessary to inform which packages will be used for the resolution, in addition to creating the object that will store the problem model (Figures 1 and 2). The last command line in Figure 2 served to create the object called ModeloMat. The model name does not necessarily have to be the same, and can be given any name. It is worth mentioning that in the future the object needs to be written in the same way, as there is a difference between upper and lower case letters. The argument in parentheses indicates that the optimizer used to solve the problem was Cbc.

```
julia> import Pkg; Pkg.add("JuMP")
julia> import Pkg; Pkg.add("Cbc")
julia> Pkg.update()
```

Figure 1 - Commands for installing and updating JuMP and Cbc packages in Julia

```
julia> using JuMP
julia> using Cbc
julia> ModeloMat = Model(with_optimizer(Cbc.Optimizer))
```

Figure 2 - Commands to load the installed packages and create the "ModeloMat" object

Then, according to Figure 3, the model's decision variables were defined with the command @variable (object name, variable name and limits, variable type). The limits can be lower, upper or both. If the type of the variable is not specified, it is considered to be real. “Bin” is used for binary variables and “Int” for integers.

```
julia> @variable(ModeloMat, x1 >= 0, Int)
julia> @variable(ModeloMat, x2 >= 0, Int)
julia> @variable(ModeloMat, x3 >= 0, Int)
julia> @variable(ModeloMat, x4 >= 0, Int)
julia> @variable(ModeloMat, x5 >= 0, Int)
julia> @variable(ModeloMat, x6 >= 0, Int)
```

Figure 3 - Defining the problem variables in Julia

As the variables present in the elaborated model are the types of snacks, the quantity produced must be whole, which justifies being classified as whole. The next step consisted of defining the objective function using the @objective command (model name, Min or Max, equation), as can be seen in Figure 4. Finally, restrictions were added using the command @constraint (model name, equation / inequality), as shown in Figure 5.

```
julia> @objective(ModeloMat, Max, 2.5x1 + 2.5x2 + 2.5x3 + 2.5x4 + 2.5x5 + 2.5x6)
```

Figure 4 - Defining the objective function of the problem in Julia

```
julia> @constraint(ModeloMat, 25x1 <= 2500)
julia> @constraint(ModeloMat, 40x3 + 75x5 + 65x6 <= 28000)
julia> @constraint(ModeloMat, 65x1 <= 6000)
julia> @constraint(ModeloMat, 20x3 <= 4000)
julia> @constraint(ModeloMat, 35x2 + 50x4 + 25x6 <= 10000)
julia> @constraint(ModeloMat, 50x4 <= 3000)
julia> @constraint(ModeloMat, 100x2 <= 4000)
julia> @constraint(ModeloMat, 65x5 + 50x6 <= 16000)
julia> @constraint(ModeloMat, x1 <= 90)
julia> @constraint(ModeloMat, x2 <= 40)
julia> @constraint(ModeloMat, x3 <= 180)
julia> @constraint(ModeloMat, x4 <= 60)
julia> @constraint(ModeloMat, x5 <= 160)
julia> @constraint(ModeloMat, x6 <= 120)
julia> @constraint(ModeloMat, x1 >= 4)
julia> @constraint(ModeloMat, x2 >= 4)
julia> @constraint(ModeloMat, x3 >= 4)
julia> @constraint(ModeloMat, x4 >= 4)
julia> @constraint(ModeloMat, x5 >= 4)
julia> @constraint(ModeloMat, x6 >= 4)
julia> @constraint(ModeloMat, 25x1 + 25x2 + 25x3 + 25x4 + 25x5 + 25x6 <= 16200)
```

Figure 5 - Defining the problem restrictions in Julia

After defining the model constraints, the command print (model name) can be used to show the entire model constructed (Figure 6).

```
julia> print(ModeloMat)
Max 2.5 x1 + 2.5 x2 + 2.5 x3 + 2.5 x4 + 2.5 x5 + 2.5 x6
Subject to
 x1 integer
 x2 integer
 x3 integer
 x4 integer
 x5 integer
 x6 integer
 x1 >= 0.0
 x2 >= 0.0
 x3 >= 0.0
 x4 >= 0.0
 x5 >= 0.0
 x6 >= 0.0
 x1 >= 4.0
 x2 >= 4.0
 x3 >= 4.0
 x4 >= 4.0
 x5 >= 4.0
 x6 >= 4.0
 25 x1 <= 2500.0
 40 x3 + 75 x5 + 65 x6 <= 28000.0
 65 x1 <= 6000.0
 20 x3 <= 4000.0
 35 x2 + 50 x4 + 25 x6 <= 10000.0
 50 x4 <= 3000.0
 100 x2 <= 4000.0
 65 x5 + 50 x6 <= 16000.0
 x1 <= 90.0
 x2 <= 40.0
 x3 <= 180.0
 x4 <= 60.0
 x5 <= 160.0
 x6 <= 120.0
```

Figure 6 - Defined model demonstrated in Julia

Finally, the command optimize! (Model name) serves to solve the problem and determine the optimal solution (Figure 7). In this step, some answers are possible: (i) Optimal, means that the problem has an optimal solution and it has been determined; (ii) Infeasible, that is, the problem is considered unfeasible because it does not have an optimal solution; and (iii) Unbounded, where the value of the objective function grows (or decreases) indefinitely as the values of the variables also increase (or decrease).

As shown in Figure 7, the problem has an optimal solution. In addition, the Cbc optimizer describes some features about solving the problem. At the top, there is the description of the optimizer. Just below, there is the comment about solving the problem by other methods. If the problem allowed continuous values in the variables, the value of the objective function would be R $ 1,609.62. The program also mentions other methods of resolution, but they were not used because they did not meet the prerequisites, as evidenced by the phrase "was tried 0 times [...]".

```
julia> optimize!(ModeloMat)
Welcome to the CBC MILP Solver
Version: 2.9.9
Build Date: Dec 31 2018

command line - Cbc_C_Interface -solve -quit (default strategy 1)
Continuous objective value is 1609.62 - 0.22 seconds
Cgl0004I processed model has 1 rows, 2 columns (2 integer (0 of which binary)) and 2 elements
Cutoff increment increased from 1e-05 to 2.4999
Cbc0012I Integer solution of -1607.5 found by DiveCoefficient after 0 iterations and 0 nodes (1.64 seconds)
Cbc0001I Search completed - best objective -1607.5, took 1 iterations and 0 nodes (2.54 seconds)
Cbc0035I Maximum depth 0, 0 variables fixed on reduced cost
Cuts at root node changed objective from -1609.62 to -1607.5
Probing was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
Gomory was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
Knapsack was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
Clique was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
MixedIntegerRounding2 was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
FlowCover was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
TwoMirCuts was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)

Result - Optimal solution found

Objective value:                1607.50000000
Enumerated nodes:               0
Total iterations:               1
Time (CPU seconds):             2.61
Time (Wallclock seconds):       2.61

Total time (CPU seconds):       2.64   (Wallclock seconds):       2.64
```

Figure 7 - Optimal solution determined by Julia

Finally, the report includes information regarding the number of iterations to achieve the optimal solution and the CPU and Wallclock times, where only the first one considers the stop times during the execution of a task. The Wallclock time counts all the time spent to perform a task, including interruptions. Thus, as can be seen, there was no interruption in the execution of the activity, since the Wallclock time was equal to the CPU time.

The results are printed using the combination of the print () command and the command you want to know. JuMP.objective_value (model name) is used for the objective function value and JuMP.value (variable) for decision variables. Therefore, according to Figure 8, the company will have a maximum revenue of R $ 1,607.50 if it decides to produce 90 drumsticks, 40 sausage rolls, 180 meat pastries, 60 ham and cheese pastries, 153 meat kibbehs and 120 cheese kibbehs .

```
julia> print(JuMP.value(x1))
90.0
julia> print(JuMP.value(x2))
40.0
julia> print(JuMP.value(x3))
180.0
julia> print(JuMP.value(x4))
60.0
julia> print(JuMP.value(x5))
153.0
julia> print(JuMP.value(x6))
120.0
```

Figure 8 - Values of the variables determined by Julia

4.2 Traveling salesman problem

In addition to defining the best mix of snacks to be produced, the company also wants to optimize its delivery route. For this, the technique used was the traveling salesman problem, which determines the shortest route to travel a series of points only once and, in the end, returns to the point of origin.

In all, the company delivers the snacks to four customers. Each client is defined as a point (or node) on the network. This, in turn, is defined as a set of nodes connected by arcs. In this particular case, the nodes represent the customers and the arcs are the distances between the customers. The origin node is C1, that is, it is the node where the journey will start (the company studied). Thus, from the focus company of the study (C1) to customer C2, the distance covered is 1.5 km. Table 4 shows the distances given in kilometers (km).

Table 4 - Distance between customers in kilometers

We we	**C1**	**C2**	**C3**	**C4**	**C5**
C1	0	1.5	1.4	2.6	3.5
C2	1.5	0	0.8	2.0	3.2
C3	1.4	0.8	0	1.5	2.2
C4	2.6	2.0	1.5	0	0.8
C5	3.5	3.2	2.2	0.8	0

It is possible to perceive that the problem is symmetrical, since the distance to travel from client i to client j is the same in the opposite way. Figure 9 represents the problem visually.

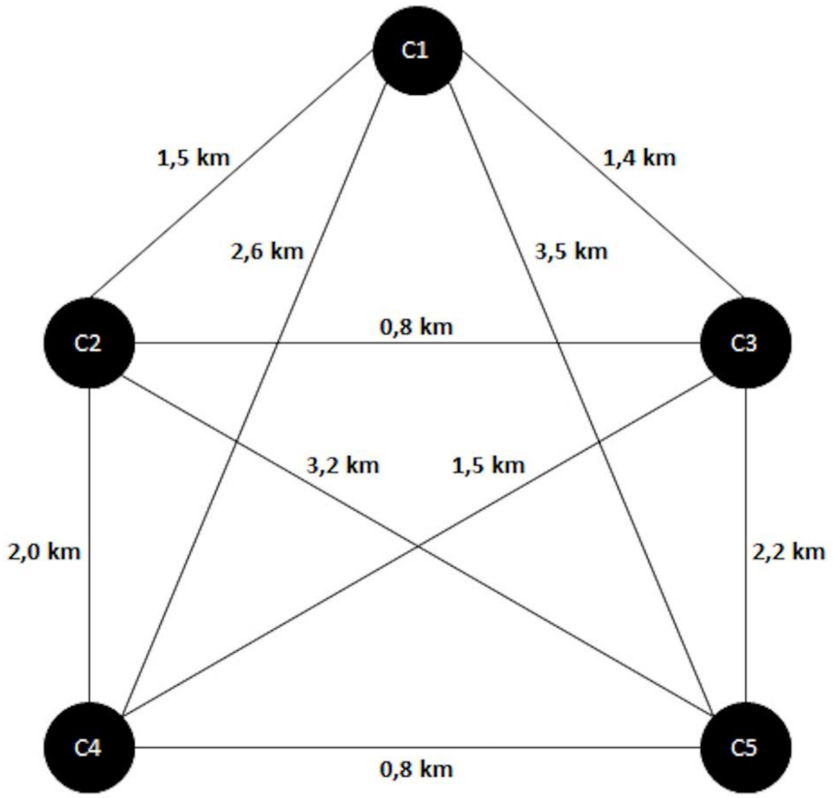

Figure 9 - Visual representation of the traveling salesman problem

Before translating the proposed problem into the Julia language, it is necessary to install the necessary packages for the resolution. Thus, the package used was "TravelingSalesmanHeuristics" (Figure 10).

```
julia> import Pkg; Pkg.add("TravelingSalesmanHeuristics")

julia> Pkg.update()
```

Figure 10 - Visual representation of the traveling salesman problem

Within the package there are several functions, such as: (i) cheapest_insertion, where the cheapest route is calculated in monetary terms; (ii) farthest_insertion, which calculates the route based on the vertices furthest from the origin vertex; and (iii) nearest_neighbor, which is the heuristic of the nearest neighbor.

After installing and updating the package, it is necessary to declare it and enter the distances between customers in square matrix format (Figure 11). Thus, the number of rows must be equal to the number of columns. Failure to complete this step will return an error in the following steps. The matrix must be stored in an object, it is worth remembering that the name of the object is free, however, it must be written in the same way in the future, since upper and lower case letters are differentiated in the language.

```
julia> using TravelingSalesmanHeuristics

julia> distclientes = [
       0 1.5 1.4 2.6 3.5;
       1.5 0 0.8 2.0 3.2;
       1.4 0.8 0 1.5 2.2;
       2.6 2.0 1.5 0 0.8;
       3.5 3.2 2.2 0.8 0;
       ]
5×5 Array{Float64,2}:
 0.0  1.5  1.4  2.6  3.5
 1.5  0.0  0.8  2.0  3.2
 1.4  0.8  0.0  1.5  2.2
 2.6  2.0  1.5  0.0  0.8
 3.5  3.2  2.2  0.8  0.0
```

Figure 11 - Declaring the package and defining the object with the distances between customers

The object created to store the distances between customers was "distclientes". The values are separated by spaces and at the end of each line, it is necessary to insert a semicolon to indicate the end of it. The matrix is started and ended with square brackets. Just below it is possible to check if the matrix was created correctly. The created matrix is of order five, has data of real type 64-bit and is of dimension 2.

Then, the corresponding heuristic function chosen to solve the problem is called. The parameters required for the function of the nearest neighbor are nearest_neighbor (distmat, firstcity). The distmat parameter corresponds to the matrix with the distances and the second parameter defines which node will be the source node. As the “distclientes” matrix has 5 lines, the parameter can receive a value from one to five, since Julia starts counting from number one (indicating the first node in the network), as seen in Figure 12.

```
julia> nearest_neighbor(distclientes,firstcity=1)
([1, 2, 3, 5, 4, 1], 7.9)
```

Figure 12 - Solution found in Julia

Therefore, analyzing Figure 12, the best route for the delivery of snacks is given by the values in square brackets. So, first you have to leave the company (node C1) and go to customer 2. Then, customer 3, go down to customer 5 and finally, go to customer 4 before returning to the origin node (C1), which is the company. The value after the comma corresponds to the total distance of the route, which is worth 7.9 km. Figure 12 illustrates the shortest path highlighted.

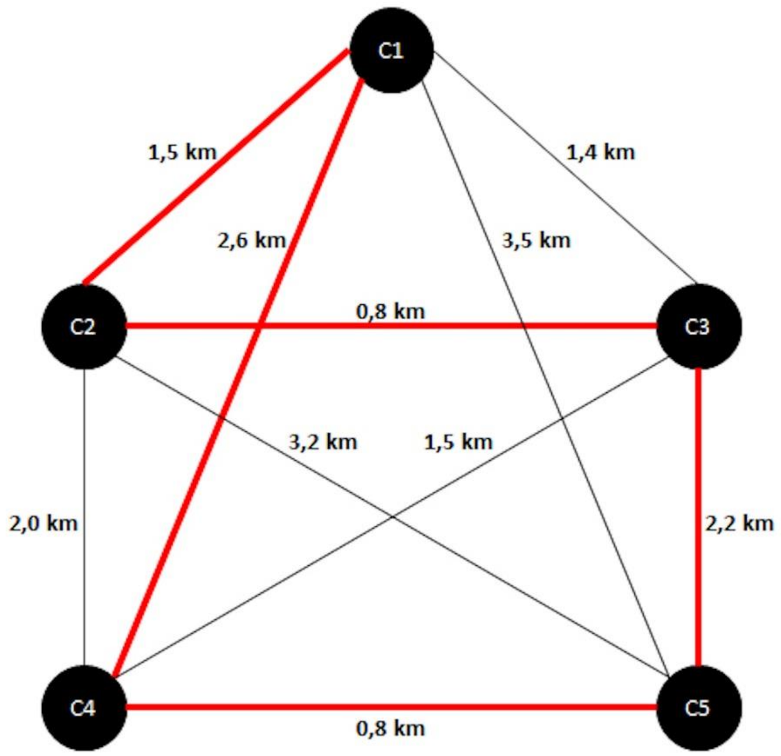

Figure 11 - Solution found in Julia

5. Final Considerations

The optimal solution to the linear programming problem that aimed at identifying the best mix of snacks to be produced to maximize the revenue of the microenterprise studied was identified. Also, the best delivery route, minimizing delivery distances between all customers, was found through the traveling salesman problem. Therefore, the objective sought in the article was achieved.

Regarding the first problem, in order for the company to obtain the maximum possible revenue, it is necessary that it keep its production close to the maximum demand, since there was a difference of only seven kibbeh of meat. Regarding the problem of the traveling salesman, it should be noted that the route for the delivery of snacks will have the same distance if it is done in the reverse way.

For both problems, the resolution method is relatively simple because the Julia programming language contains specific functions and packages for each. Consequently, Julia takes advantage of different programming languages to achieve efficiency and productivity. It also stands out the fact that it is a free and open source language. Still, it should be noted that the article presents two of the various optimization techniques that are available for resolution in the Julia language. The fact that it brings together several techniques helps in the convenience for analysts, as they can specialize only in language and solve various business problems.

As for the contribution in the business sphere, this article can be used as a tutorial for using these operational research techniques in micro and small companies, since the problem resolutions were made step-by-step so that the reader can use them. them. The use of the Julia language was considered useful and possible to be applicable in small companies. As for the academic field, this article can be used as a tutorial for future projects and research that include linear programming and / or the traveling salesman problem. In addition, it discloses practical application of PO problems in micro companies, increasing the disclosure that a complex area can be used for managerial support also in micro and small companies.

It is suggested, for future research, the application of the Julia language in other micro and small companies in order to obtain businessmen's feedback regarding the experience of use and perceived management benefits after implementation of the results achieved.

Bibliographic references

ALMEIDA, WS; SILVA, ME; SANTOS, NVM; RIBEIRO, LM; BACHEGA, SJ Application of entire linear programming to maximize the profit of a company in the beauty and aesthetics sector. In: NATIONAL MEETING OF PRODUCTION ENGINEERING, 37., 2017, Joinville. Anais .... Rio de Janeiro: ABEPRO, 2017. p. 01-16.

ANDERSON, TA, LIU, H., KUPER, L., TOTONI, E., VITEK, J .; SHPEISMAN, T. Parallelizing Julia with a Non-Invasive DSL. DARTS. v. 3, n. 2, p. 01-29, 2017.

ARENALES, M .; MORABITO, R .; ARMENTANO, V .; YANASSE, H. Operational Research. 1. ed. Rio de Janeiro: Elsevier, 2007.

BALDASSI, C. A method to reduce the rejection rate in Monte Carlo Markov chains. Journal of Statistical Mechanics: Theory and Experiment. Vol. 2017, pp. 01-19, 2017.

BEZANSON, J .; EDELMAN, A .; KARPINSKI, S .; SHAH, VB Julia: A Fresh Approach to Numerical Computing. SIAM Review, Vol. 59, n. 1, pp. 65-98, 2017.

BEZANSON, J .; KARPINSKI, S .; SHAH, V .; EDELMAN, A. Why we Created Julia. 2012. Available at:http://julialang.org/blog/2012/02/why-we-created-julia. Accessed on 10/09/2019.

BRYMAN, A. Research methods and organization studies. London: Uniwin Hyman, 1989.

CARVALHO, MCM de. The construction of scientific knowledge: some propositions. In: CARVALHO, MCM de (org.). Building knowledge. 2.ed. Campinas: Papirus, 2000.

CASTELLUCCIA, PB JULIA AND JuMP: NEW TOOLS FOR PROGRAMMING. Operational Research for Development, v. 9, n. 2, p. 48-61, 2017.

CRESWELL, JW Research project. Qualitative, quantitative and mixed methods. 3. ed. Porto Alegre: Artmed, 2010.

EHRLICH, PJ Operational Research. 5. ed. São Paulo: Atlas, 1985.

GIL, AC How to prepare research projects. 4. ed. São Paulo: Atlas, 2009.

HILLIER, FS; LIEBERMAN, GJ Introduction to Operations Research. 9. ed. Porto Alegre: AMGH, 2013.

JULIALANG.ORG. Julia Micro-Benchmarks. 2018. Available at:https://julialang.org/benchmarks/. Accessed on 03/28/2019.

KARP, RM On the computational complexity of combinatorial problems. Newtorks 5, 1975.

LINS, MPE; CALÔBA, Guilherme M. Linear Programming. 4. ed. Rio de Janeiro: Interciência, 2010.

MOGENSEN, PK; RISETH, AN Optim: A mathematical optimization package for Julia. Journal of Open Source Software, v. 3, n. 24, p. 01-03, 2018.

MOREIRA, DA Operacional Research - Introductory course. 2. ed. São Paulo: Cengage Learning, 2010.

PASSOS, EJPF Linear Programming as an instrument of operational research. 1. ed. São Paulo: Atlas, 2008.

PEREIRA, CD; CUNHA, GF; SILVA, MG Simulation in operational research: a literary review. EEPA, Campo Mourão. Available in:http://www.fecilcam.br/anais/ix_eepa/data/uploads/3-pesquisa-operacional/3-03.pdf. Accessed on 10/10/2019.

PRADO, Darci. Linear Programming. 5. ed. Belo Horizonte: INDG, 2007.

RACKAUCKAS, C .; SCHILLING, T .; NIE, Q. Mean-Independent Noise Control of Cell Fates via Intermediate States. IScience, v. 3, pp. 11-20, 2018.

SANTANA, TMN; GUIMARAES, RRS; FONSECA, GS; GONCALVES, LF; BACHEGA, SJ Application of the problem of the minimum generating tree in an internet provider company. In: PRODUCTION ENGINEERING SYMPOSIUM, 2018, Catalan. Anais ... Catalan: UFG-RC, 2018. p. 01-09.

SILVEIRA, JFP Traveling salesman problem. UFRGS, Rio Grande do Sul. Available at:http://www.mat.ufrgs.br/~portosil/caixeiro.html. Accessed on 17/10/2018.

SOUSA, LT; VEIGA, AG; CHAVES, MFC; VAZ, MV; BACHEGA, SJ Application of network optimization in a company in the poultry sector. In: MACHADO, MWK (Org.). Production Engineering: What's Your Plan? Ponta Grossa: Atena Editora, 2019, p. 356-367.

SOUZA, LC; VIANA, GG; YOSHIDA, YM; TELLAROLI, LF; BACHEGA, SJ The shortest path problem applied to a newspaper company. In: POISSON (Org.). Production Management in Focus - Volume 26. Belo Horizonte: Poisson, 2019, v. 26, p. 127-135.

TAHA, HA Operational Research. 8. ed. São Paulo: Pearson Pretince Hall, 2008.

TEIXEIRA, NB; SILVA, CF; VIRGOLINO, GA; SOUZA, A .; BACHEGA, SJ Application of the entire linear programming in a small precast barbecue factory. In: PRODUCTION ENGINEERING SYMPOSIUM, 24., 2017, Bauru. Anais ... Bauru: UNESP, 2017. p. 01-12.